UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADO EN INGENIERÍA DEL SOFTWARE

MONITORIZACIÓN Y RECOMENDACIÓN DE EJERCICIO FÍSICO A TRAVÉS DE PULSERA/RELOJ INTELIGENTE

MONITORING AND RECOMMENDATION OF PHYSICAL EXERCISE TROUGH SMART WATCH

Realizado por
MIGUEL PARDAL MARTÍN

Tutorizado por
EDUARDO GUZMAN DE LOS RISCOS

Departamento
LENGUAJE Y CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO 2021

Resumen

Durante los últimos años, existe un creciente número de dispositivos ideados para registrar distintos indicadores personales en lo relacionado con la salud física. Prácticamente, cualquier persona o cuenta con uno de estos relojes/ pulseras inteligentes o conoce a alguien que lo usa en su día a día. Estos dispositivos están totalmente integrados en la vida diaria y cada vez cuentan con más funcionalidades.

El problema surge ante esta gran variedad de dispositivos, marcas y Sistemas Operativos. Cada empresa cuenta son su propio software, lo que supone un problema al cambiar de dispositivos, puesto que hay que instalar nuevas aplicaciones en el móvil, nuevos registros, etc. La idea de este proyecto es unificar todos estos pasos y tener una aplicación móvil universal que, independientemente del móvil, Sistema Operativo o pulsera/ reloj inteligente, podamos usar ante cualquier cambio. Además, al realizar este cambio, los datos recogidos anteriormente no se perderán puesto que estarán almacenados de manera que se puedan consultar en cualquier momento.

Por último, dado que el sistema recogerá estos datos, se podrán ir realizando recomendaciones de entrenamientos y ejercicio físico en función de estos datos recogidos, aunque bien es cierto que esta actividad física debe estar supervisada por un especialista. El sistema solamente hará recomendaciones al usuario, pero este será quién tomará la decisión final.

PALABRAS CLAVE

Aplicación móvil, Flutter, Smartwatch, Actividad Física, Salud

Abstract

During last years, there has been a growing number of devices designed to record different personal indicators related to physical health. Almost everyone either owns one of these smartwatches or knows someone who uses it on a day-to-day basis. These devices are fully integrated into daily life and increasingly have more functionalities.

The problem arises from this great variety of devices, brands and Operating Systems. Each company has its own Software, which is a problem when changing devices, since new applications have to be installed on the mobile, new registrations, etc. The idea of this project is to unify all these steps and have a universal mobile application that, regardless of the mobile, Operating System or bracelet / smart watch, we can use before any change. In addition, by making this change, the data previously collected will not be lost since it will be stored so that it can be consulted at any time.

Finally, since the system will collect this data, training and physical exercise recommendations can be made based on these data collected, although it is true that this physical activity must be supervised by a specialist. The system will only make recommendations to the user, but this will be the one who will make the final decision.

KEYWORDS

Mobile application, Flutter, Smartwatch, Physical Activity, Health

Índice

1

Introducción

1.1 Motivación

El presente proyecto surge de la gran variedad de aplicaciones destinadas a los usuarios a la hora de cuantificar tanto la actividad física como otros datos de salud (ritmo cardíaco, capacidad aeróbica, …). La idea es, en la medida de los posible, no depender de una aplicación de cada empresa por cada pulsera/reloj inteligente, sino unificar la recogida de esa información en una única aplicación, independientemente del Sistema Operativo del dispositivo móvil (Android o iOS), así como del tipo de pulsera/ reloj inteligente. Así mismo, a partir de esa recogida de datos, desde la propia aplicación se realizarán recomendaciones de entrenamientos o ejercicios, aunque estos deberían seguir la supervisión de un experto.

1.2 Objetivos

El objetivo principal es desarrollar una aplicación móvil multiplataforma (Android e iOS) que sea capaz de recoger información de la actividad física de una persona a través de una pulsera/ reloj inteligente y poder realizar recomendaciones de

ejercicio físico o entrenamientos. Además, esta información se guardará de manera remota por dos motivos. El primero de ellos sería para poder hacer un seguimiento del usuario, y determinar su evolución. El segundo, sería la posibilidad de guardar estos datos en la nube y asociarlos a un perfil de usuario, de manera que, si se inicia sesión en otro dispositivo o se cambia de pulsera/ reloj inteligente, los datos anteriores sigan presentes.

1.3 Tecnologías y herramientas principales utilizadas

Las tecnologías que se han decidido usar para este proyecto están elegidas por motivos de adaptabilidad a los requisitos de este. A continuación, se expondrán ordenadas según el componente donde se usan, así como una descripción de esta y la justificación correspondiente de su uso.

1.3.1 Tecnologías y herramientas utilizadas en el sistema de persistencia

- MongoDB [1]: base de datos distribuida tipo No Relacional, basada en documentos y de uso general que ha sido diseñada para desarrolladores de aplicaciones modernas y para la era de la nube. Es el sistema de persistencia dónde se almacenan todos los datos recogidos en el proyecto. Los principales motivos de su elección son su velocidad procesando volúmenes de datos y su adaptabilidad.
- MongoDB Compass [2]: interfaz gráfica para interactuar de manera directa con MongoDB. Permite explorar visualmente los datos, así como ejecutar consultas en segundos. De igual manera, puede visualizarse el rendimiento de cada consulta, así como optimizar las mismas. Su sencillez de uso y la posibilidad de visualizar los datos de manera rápido y sencilla, son los motivos para su uso.

1.3.2 Tecnologías y herramientas utilizadas en el lado del servidor

- JavaScript: lenguaje de programación orientado a objetos y eventos. Es el lenguaje principal usado en el proyecto para el lado del servidor. El dialecto del lenguaje usado es ECMAscript 2015 (ES6 [3]). Su simplicidad y las nuevas funcionalidades han sido los principales motivos para su elección. Cabe mencionar que TypeScript, lenguaje precompilado, fue otra de la opción a la hora de desarrollar el servidor, pero se optó finalmente por JavaScript por su facilidad de integración.
- Node.js [4]: entorno de ejecución de código JavaScript orientado a eventos asíncronos. Node.js está diseñado para crear aplicaciones escalables.
- Npm [5]: gestor de paquetes JavaScript.

1.3.3 Tecnologías y herramientas utilizadas en el lado del cliente

- Flutter [6]: kit de herramientas de UI de Google para realizar aplicaciones, compiladas nativamente, para móvil, web y escritorio desde una única base de código. Esta tecnología tiene su propia sección más adelante comentando sus detalles, así como los motivos de su elección frente a otras tecnologías.

1.3.4 Otras Tecnologías y herramientas utilizadas

- git [7]: controlador de versiones del proyecto.
- GitHub [8]: servicio de almacenamiento en la nube de repositorios git.
- CodeFactor [9]: software que permite evaluar la calidad del código de cada repositorio.
- Es-lint [10]: librería de Visual Studio Code para marcar errores de estilos según el estándar de ECMAScript.

- Firebase [11]: plataforma para el desarrollo de aplicaciones tanto web como móviles. Se usará para la gestión de acceso de los usuarios y el almacenamiento de documentos del usuario (imagen de perfil y entrenamientos).
- Postman [12]: software que permite realizar llamadas en el protocolo HTTP de manera sencilla e intuitiva. Durante la fase de desarrollo, facilita realizar las consultas sobre la aplicación en el lado del servidor.
- Visual Studio Code [13]: editor de código multiplataforma y multilenguaje. Utilizado para el desarrollo completo del proyecto, tanto en el lado del servidor como en el del cliente.
- Iphone 7: dispositivo móvil para probar la ejecución de la aplicación así como conectar las pulseras/ relojes inteligentes. La versión del sistema operativo es iOS 14.4.2.
- AppleWatch Series 3: reloj inteligente principal usado para monitorizar al usuario y recopilar los datos pertinentes.
- Xiaomi Miband 5: pulsera inteligente secundaria usada para monitorizar al usuario y recopilar los datos pertinentes.

1.4 Estudio de mercado

En este apartado, se pretende abordar la gran variedad de aplicaciones que hay en el mercado (Play Store y App Store), tanto de marcas conocidas para sus propios dispositivos como de marcas externas. De esta manera, queda de manifiesto la utilidad de esta aplicación con la idea de unificar esta recogida de datos en un solo punto de entrada común e independiente tanto del dispositivo como del Sistema Operativo del móvil.

- Adidas Runtastic → conocida antiguamente solo como Runtastic, es la aplicación móvil para entrenar más conocida y puede que la que inició el auge

del uso de los relojes/ pulseras inteligentes. Permite recoger información variada como entrenamientos de andar, correr y otras modalidades deportivas. Incluye un reproductor de música así como capacidad de compartir datos con otros usuarios.

- Nike Training Club → aplicación que incluye rutinas de entrenamiento que se pueden realizar en cualquier entorno y sin necesidad de contar con material específico o difícil de conseguir.
- GoogleFit → aplicación propia de Google que sincroniza los datos recogidos con los datos personales del usuario. Más simple que las anteriores, pero con la potencia del motor Google.
- Xiaomi Mi Fit → al igual que Google, es la aplicación propia de la marca Xiaomi. Cuenta con integración completa de sus dispositivos, pero esa es su limitación. Incompatible con relojes/ pulsera inteligentes de otras marcas.
- Runkeeper → esta aplicación registra los datos del usuario cuando corre. Permite establecer objetivos, reproducción de música y escuchar estadísticas en tiempo real.

1.5 Estructura de la memoria

En esta sección, se han establecido cuáles son los objetivos de este TFG así como las tecnologías y herramientas que se han utilizado en su desarrollo. También se ha realizado un estudio de herramientas similares a la desarrollada en este TFG existentes en el mercado.

En esta sección se han establecido tanto la motivación como los objetivos principales del proyecto; se han explicado las principales tecnologías y herramientas usadas en el TFG, y a continuación, se ha realizado un estudio de aplicaciones similares

que estuvieran disponibles en los dos principales Sistemas Operativos del mercado (Android e iOS) y evaluar qué ofrecen cada una de ellas.

Una vez finalizado el estudio de mercado, se procederá a explicar el proceso de Ingeniería de Requisitos y determinar los Casos de Uso de la aplicación. De igual manera, se describirán los primeros prototipos de la aplicación tanto para la interfaz de usuario como para el desarrollo del servidor.

Tras terminar la fase de Ingeniería de Requisitos, se abordará el proceso de desarrollo propio. Durante esta fase, se desarrollaron los prototipos elaborados en la fase anterior, solventando diversas dificultades y errores que aparecieron.

Por último, se explicarán las pruebas realizadas en el marco de este TFG consistentes en ceder la aplicación a diversos usuarios con distintos móviles y relojes/ pulseras inteligentes para que la probaran en un entorno real.

2

Flutter

2.1 ¿Qué es Flutter?

Flutter es el kit de herramientas de UI de Google para realizar aplicaciones, compiladas nativamente, para móvil, web y escritorio desde una única base de código. Además, ofrece un desarrollo rápido, con la capacidad de crear interfaces de usuario flexibles y con rendimiento nativo. Flutter permite desarrollar aplicaciones tanto para el Sistema Operativo Android como iOS ahorrando tiempo a la hora de desarrollar estas aplicaciones para ambos sistemas operativos. Además, también soporta el desarrollo web y, a partir de la versión 2.0, se incluye el desarrollo de aplicaciones de escritorio para Windows, macOS y Linux.

Flutter [14] nació de manera oficial en 2017 bajo el nombre *Sky* y solo estaba disponible bajo el Sistema Operativo de Android. En 2018, la primera versión estable fue lanzada. En 2020, con la versión 1.17.0 mejoraron de manera significativa el rendimiento en aplicaciones desarrolladas para iOS (hasta un 50%). Por último, en

2021 se lanza la versión 2.0, que incluye la posibilidad del mencionado desarrollo multiplataforma.

El lenguaje utilizado para desarrollar aplicaciones en Flutter es Dart [15], un lenguaje de programación desarrollado por Google e influenciado por C#, JavaScript y Java que pretende ofrecer una alternativa al desarrollo web.

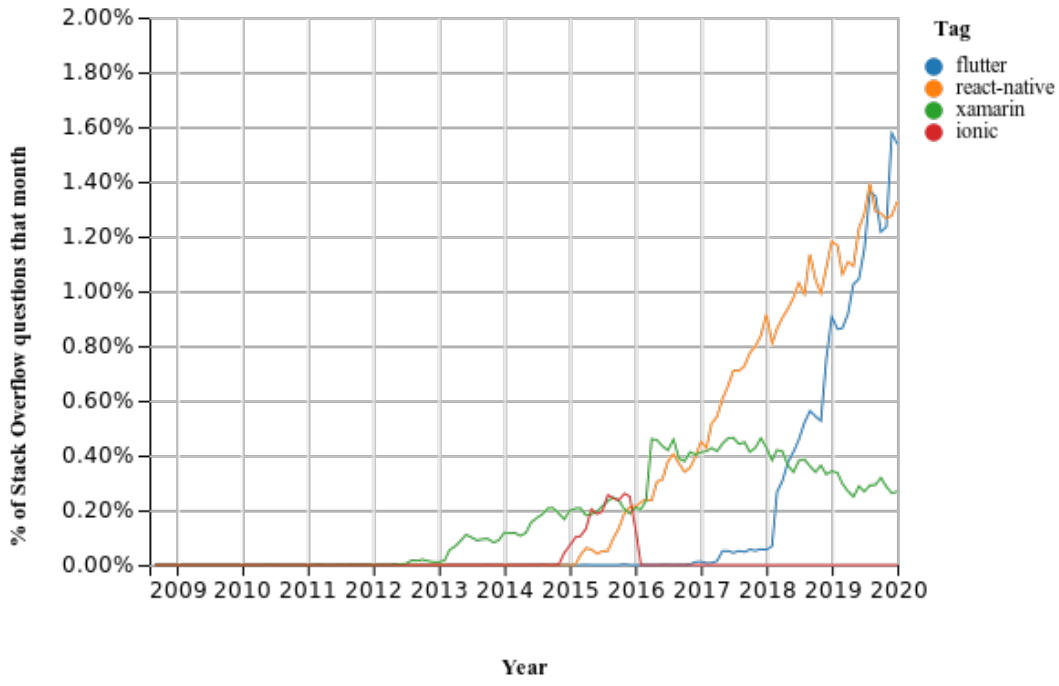

Ilustración 1. Evolución de consultas en Stack Overflow. Fuente: BBVA Next Technologies

Como puede verse en la Ilustración 1, el crecimiento de las consultas en el portal Stack Overflow en los últimos años, y desde su nacimiento, es exponencial, superando a su principal competidor (React-Native).

2.2 Flutter Vs React Native

Dado que React Native [16] y Flutter comparten nicho de mercado, es necesario realizar una pequeña introducción de lo que es React Native y una comparación entre ambos marcos de desarrollo.

React Native es un framework de Javascript desarrollado por Facebook que permite desarrollar aplicaciones para Android e iOS y que además permite una implementación nativa. En la Tabla 1, se puede visualizar una comparación entre ambas herramientas.

	Flutter	React Native
Lanzamiento oficial	Diciembre 2018	Marzo 2015
Desarrollador	Google	Facebook
Open Source	Si	Si
Lenguaje de programación	Dart	Javascript
Popularidad	118000 estrellas en GitHub (abril 2021)	94700 estrellas en GitHub (abril 2021)
Hot reload	Sí	Sí
Rendimiento nativo	Excelente	Excelente
Interfaz de Usuario	Las aplicaciones se ven bien en versiones actuales como anteriores del Sistema Operativo. Se comportan y se ven de manera similar tanto en	Los componentes se parecen a los nativos de cada Sistema Operativo. A través de bibliotecas de terceros, pueden usarse componentes para que la aplicación se vea idéntica

	Android como iOS, aunque pueden adaptarse a cada Sistema Operativo. Igualmente se comportarán de manera nativa.	en cualquier versión del dispositivo.
Multiplataforma	iOS, Android, Web y Aplicaciones de escritorio	iOS, Android, Web y Aplicaciones de escritorio
Aplicaciones populares	Alibaba, Google Ads, BMW, Sonos	Instagram, Facebook, Skype, Tesla
Ventajas	Rápido crecimiento de la comunidad y popularidad; buena documentación oficial; desarrollo multiplataforma; magnificas interfaces de usuario gracias a los widgets	Estabilidad (6 años de maduración); amplia comunidad; variedad de aplicaciones; fácil de aprender; cantidad de recursos para ayudar en el desarrollo
Desventajas	El diseño de la aplicación es exclusivo para el Sistema Operativo; interacción directa con el Sistema Operativo	Comunicación Bluetooth con otros dispositivos; no adecuado para aplicaciones con funcionalidad muy específica

Tabla 1. Comparativa Flutter - React Native

2.3 Funcionamiento de Flutter

Flutter, a través de Dart, produce código nativo para todas las plataformas y muy cercanas al rendimiento de los dispositivos móviles con chips arm64. Como se puede ver en la Ilustración 2, la aplicación desarrollada con Flutter conecta directamente con la plataforma de destino, evitando compilar el código en cada uso y mejorando el rendimiento de esta.

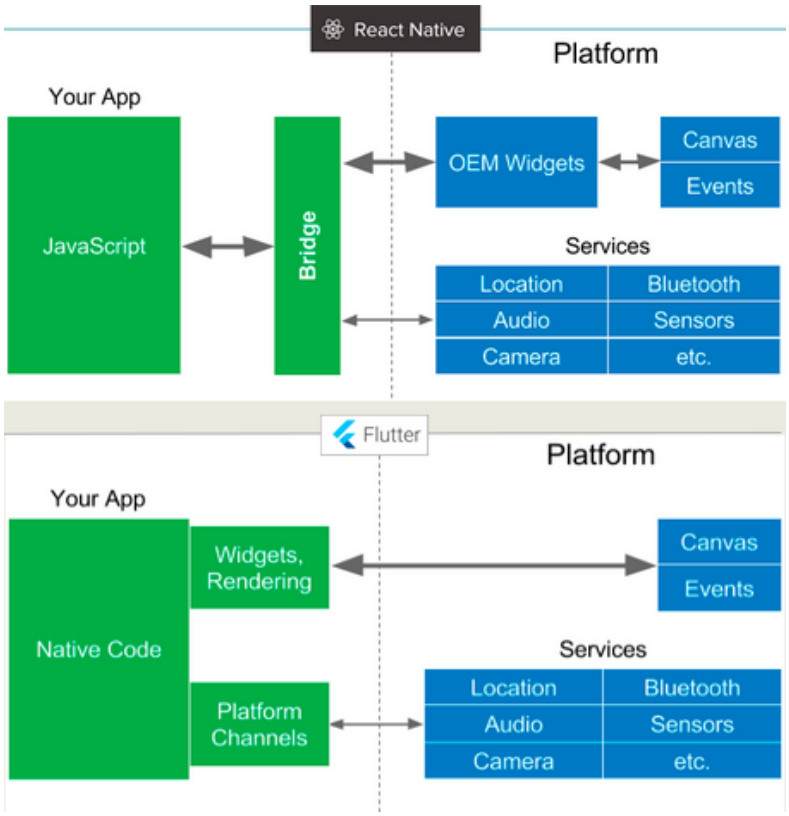

Ilustración 2. Rendimiento de Flutter

2.3.1 El árbol de widgets

Los Widgets describen cómo debería ser la vista de la aplicación, dada su configuración y estado actuales. Cuando el estado de un widget cambia, el widget reconstruye su descripción, determinando los cambios mínimos necesarios en el árbol de renderizado subyacente para la transición de un estado al siguiente.

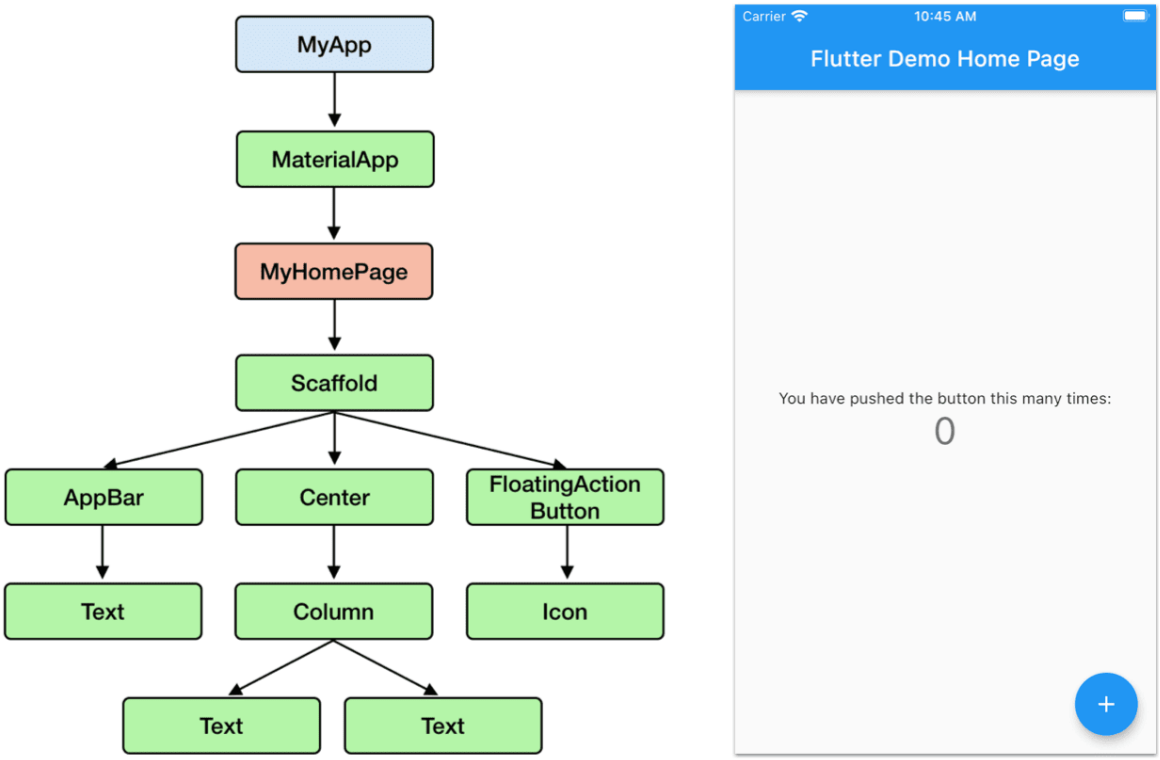

Ilustración 3. Ejemplo del Árbol de Widgets

Cada rectángulo representa un widget, y como se puede observar, un widget puede contener widgets en su interior.

2.3.2 Stateless Widget

Se tratan de widget sin estado o estáticos. No guardan ningún tipo de estado, como su nombre indica, ni tienen valores en su interior que puedan cambiar. Un ejemplo puede ser el widget *Image* que muestra una imagen fija.

2.3.3 Statefull Widget

Widgets con estado o dinámicos. Estos widgets realizan un seguimiento del estado y modifican su interfaz en función de los cambios realizados. Un *Button* podría ser un widget de este tipo que al pulsarlo cambie su estado.

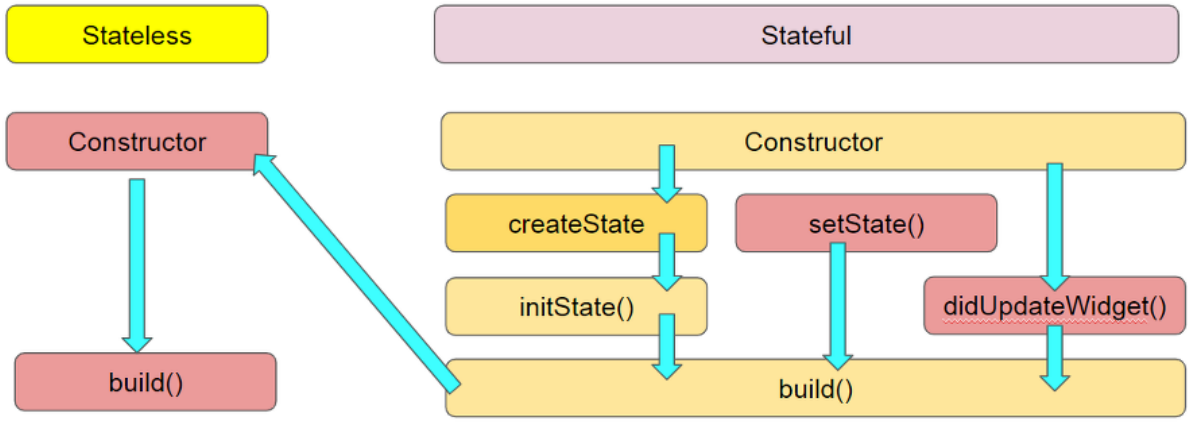

Ilustración 4. Ciclo de vida de un Widget

3

Ingeniería de Requisitos

Una vez que se ha puesto en contexto la principal herramienta para el desarrollo de la aplicación, da comienzo la etapa donde se determinan los requisitos principales del proyecto. Utilizando el lenguaje UML, se elaboran diversos diagramas para explicar y detallar su funcionamiento.

Los requisitos de la aplicación giran en torno a la eficiencia y eficacia de la aplicación de cara al usuario final. Al tratarse de una aplicación de uso cotidiano y diario orientada a cualquier público, la sencillez de esta tiene que ser un punto fundamental.

El actor principal y único de la aplicación será el usuario que haga uso de ella, por tanto, no se especificará este detalle en los Casos de Uso puesto que siempre será el mismo.

3.1 Casos de Uso

Se mostrarán a continuación los Casos de Uso generales de la aplicación.

Caso de Uso ID	#1
Caso de Uso	Registro de un nuevo usuario
Descripción	El sistema permitirá el registro de un nuevo usuario
Precondición	• El usuario no habrá iniciado sesión • El usuario no estará registrado en el sistema anteriormente
Postcondición	El usuario estará registrado en el sistema

Caso de Uso ID	#2
Caso de Uso	Inicio de sesión en el sistema
Descripción	El usuario será capaz de iniciar sesión con su perfil en la aplicación
Precondición	• #1
Postcondición	El usuario habrá iniciado sesión en el sistema y accederá a la pantalla de inicio de la aplicación

Caso de Uso ID	#3
Caso de Uso	Edición datos personales
Descripción	El usuario será capaz de modificar los datos personales recogidos en el sistema
Precondición	• #2
Postcondición	El usuario habrá sido capaz de modificar sus datos personales

Caso de Uso ID	#4
Caso de Uso	Búsqueda de dispositivos
Descripción	El sistema se encargará de escanear dispositivos para conectarlos mediante conexión Bluetooth y mostrarlos mediante un listado
Precondición	• #2 • La conexión Bluetooth del dispositivo tiene que estar encendida • El usuario debe conceder permisos al dispositivo para acceder al escaneo de dispositivos Bluetooth
Postcondición	El sistema mostrará un listado de dispositivos Bluetooth a los que es posible conectarse

Caso de Uso ID	#5
Caso de Uso	Mostrar datos
Descripción	El sistema mostrará los diferentes datos almacenados en el sistema de persistencia. También se mostrará la información del usuario
Precondición	• #2 • Navegación por la aplicación • Conexión a internet • Datos sincronizados con el sistema de persistencia
Postcondición	El sistema mostrará los datos del usuario

Caso de Uso ID	#6
Caso de Uso	Actualización de datos

Descripción	El sistema sincronizará los datos recogidos del dispositivo Bluetooth en el sistema de persistencia.
Precondición	• #2 • #4 • Conexión a internet
Postcondición	#5

Caso de Uso ID	#7
Caso de Uso	Completar entrenamiento
Descripción	El usuario será capaz de marcar el entrenamiento recomendado como completado
Precondición	• #2 • Conexión a internet
Postcondición	• #5 • El sistema recomendará un nuevo entrenamiento a completar

Caso de Uso ID	#8
Caso de Uso	Compartir en Redes Sociales
Descripción	El sistema permitirá compartir los datos en las Redes Sociales
Precondición	• #2 • Conexión a internet • El usuario deberá iniciar sesión en la Red Social de destino
Postcondición	• #5

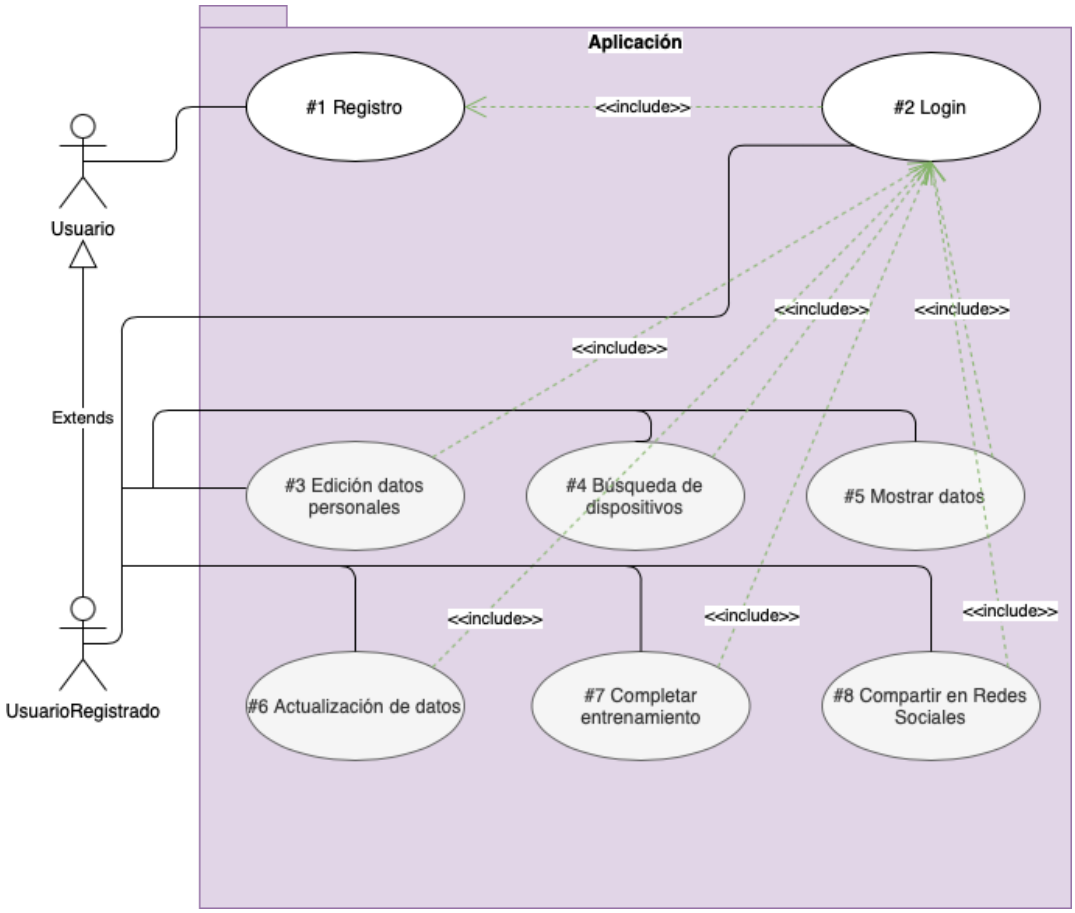

Ilustración 5. Diagrama Casos de Uso

3.2 Requisitos funcionales

Los siguientes requisitos funcionales parten de la base de que el usuario ya ha accedido a la aplicación e iniciado sesión en la misma.

Nombre	Escaneo de dispositivos Bluetooth
Descripción	La aplicación permitirá escanear dispositivos Bluetooth a los que se pueda conectar

Nombre	Personalización de la información del usuario
Descripción	La aplicación permitirá que el usuario modifique sus datos personales en cualquier momento

Nombre	Personalización de las preferencias/ objetivos de entrenamiento del usuario
Descripción	La aplicación permitirá que el usuario modifique sus preferencias/ objetivos de entrenamiento en cualquier momento

Nombre	Visualizar datos en forma de gráficos
Descripción	La aplicación permitirá que el usuario visualice la información relativa a los datos almacenados en el sistema de persistencia a través de gráficos

Nombre	Mostrar errores
Descripción	La aplicación mostrará los errores que ocurran

Nombre	Completar entrenamientos
Descripción	La aplicación permitirá al usuario marcar cuando un entrenamiento ha sido completado y recomendará uno nuevo

Nombre	Compartir en Redes Sociales
Descripción	La aplicación permitirá al usuario compartir en Redes Sociales sus entrenamientos completados

3.3 Requisitos no funcionales

Nombre	Escaneo de dispositivos
Descripción	El sistema escaneará en búsqueda de dispositivos Bluetooth durante 4 segundos, pasado ese tiempo habrá que indicar a la aplicación que vuelva a escanear

Nombre	Fiabilidad
Descripción	Al ser un sistema joven, que depende de otros dispositivos para su correcto funcionamiento, la fiabilidad se sitúa en el 95%

Nombre	Compatibilidad
Descripción	La aplicación será compatible con móviles iPhone con un Sistema Operativo superior a la versión 12.0 y con móviles con Sistema Operativo Android superior a 5.0

Nombre	Eficiencia
Descripción	La aplicación no consumirá mucha batería, dado que la sincronización de datos con el dispositivo Bluetooth solo ocurrirá cuando esta se inicie o cuando el usuario así lo ejecute

Nombre	Usabilidad
Descripción	El usuario será capaz de recorrer la aplicación y obtener la información buscada en el menor número de interacciones posible. La aplicación estará disponible en idioma español e inglés

4

Modelado y Diseño de Software

4.1 Diagrama de clases

Tras realizar el análisis de requisitos del sistema, se procede a documentar el diagrama de clases de este. En él, se representa la conexión de los elementos de la aplicación, así como su conexión en la base de datos. No se trata de un modelo de la base de datos, puesto que esta es no relacional.

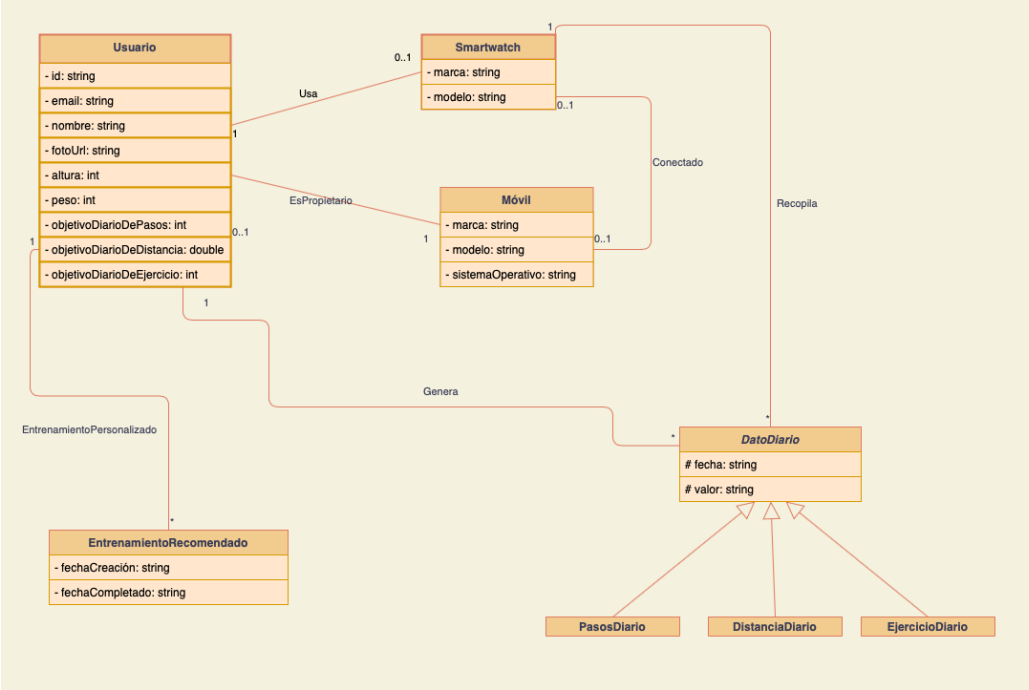

Ilustración 6. Diagrama de clases

4.1.1 Usuario

La clase Usuario es el eje principal de toda la aplicación. Como bien se menciona durante este documento, la idea de la aplicación es de que un usuario pueda usarla independientemente del dispositivo móvil y del reloj/ pulsera inteligente, por lo que es la entidad base.

Se recogerá información básica y necesaria para el correcto funcionamiento de la aplicación. Para el registro de la aplicación, que será totalmente gratuito, se solicitará el *email*, deberá ser único en el sistema, y una *contraseña*. El resto de los datos que se incluyen ayudarán para mejorar la experiencia del usuario.

Esta clase, además, cuenta con 4 relaciones con otras clases:

- Usa → con la clase Smartwatch
- EsPropietario → con la clase Móvil
- Genera → con la clase DatoDiario

- EntrenamientoPersonalizado → con la clase EntrenamientoRecomendado

4.1.2 Smartwatch

Esta clase representa el reloj/ pulsera inteligente que el usuario usa diariamente. Este dispositivo es de una *marca* y un *modelo* en concreto que son sus atributos de clase.

Sus relaciones con otras clases son las siguientes:

- Recopila → con la clase abstracta DatoDiario
- Conectado → con la clase Móvil

4.1.3 Móvil

La clase Móvil representa el dispositivo móvil del usuario. Es imprescindible que el usuario cuente con un móvil para poder utilizar la aplicación. La instancia de esta clase recopilará la información de la *marca*, *modelo* y *sistemaOpertivo*.

4.1.5 DatoDiario

Esta clase recopila información diaria básica con los atributos *fecha* y *valor*. De ella, heredan las clases PasosDiario, DistanciaDiario y EjercicioDiario, para diferenciar el tipo de información básica que se recoge.

4.1.6 EntrenamientoRecomendado

Esta clase es la responsable de mostrar los entrenamientos recomendados al usuario de la aplicación en función de la información recopilada y procesada. Sus atributos son la *fechaCreación* y *fechaCompletado*.

4.2 Arquitectura

Para la programación y arquitectura, tanto de la parte del servidor como la aplicación móvil, se ha tomado como referencia el libro "Clean Code" [17], del que se puede destacar algunas recomendaciones como usar variables descriptivas por encima de comentarios o usar "Getters" y "Setters" en lugar de acceder a las propiedades/atributos de cada modelo o clase.

De entre todas las arquitecturas para el diseño del software, se ha decidido implementar la arquitectura por capas debido a ser un patrón muy versátil y que además se integra bastante bien en el desarrollo de esta aplicación.

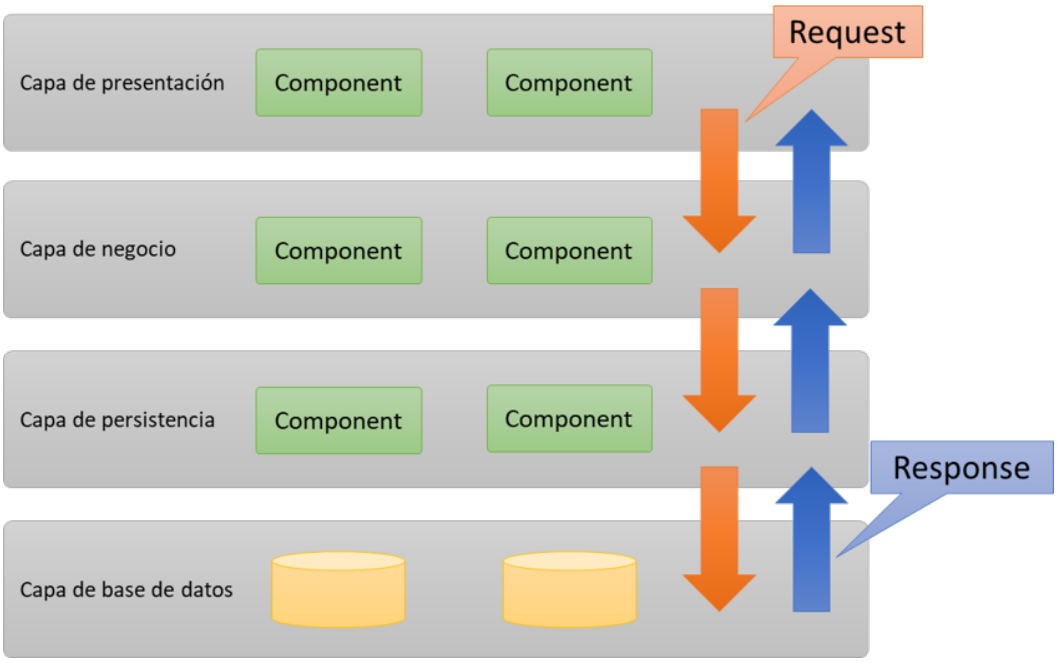

Ilustración 7. Arquitectura por capas. Fuente: https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/capas

Como se puede ver en la Ilustración 7, cada capa se conecta con la inmediatamente superior o inferior. Desde las capas superiores, se realizan las

peticiones, mientras que las respuestas se envían desde las capas inferiores hacia las superiores.

En el caso de esta aplicación, la *Capa de presentación* vendría a ser la aplicación móvil, desde dónde el usuario iniciará todas las peticiones y la manera en la que se podrá interactuar con el resto del sistema. La *Capa de negocio* y la *Capa de persistencia* serían la aplicación desarrollada en el servidor. En la primera es dónde se realizará toda la lógica de negocio y validaciones. Por ejemplo, todas las operaciones y llamadas a *"endpoints"* están restringidas a que en la cabecera de la petición exista un token válido (en el apartado 4.2.2 se desarrollará esta idea en detalle). En la *Capa de persistencia* es donde se produce la comunicación directa con la base de datos. Por último, la *Capa de base de datos* es la propia base de datos.

La idea que recoge este patrón es la independencia de cada capa, de manera que, si en un futuro se desea reescribir la aplicación móvil, por ejemplo, no sea necesario modificar nada de las capas subyacentes.

4.2.1 Aplicación Flutter

Para el desarrollo de la aplicación móvil, se ha decidido seguir el patrón BLoC. El patrón BLoC es un patrón que sirve para la gestión de estados, altamente recomendado por los desarrolladores de Google. Se trata de un patrón que permite además el acceso a diferentes datos en varias partes de la aplicación.

Cada widget debe tener su propio BLoC que cuenta con 3 principales ventajas:

- Centralizar la lógica de negocio: de esta manera se pretende extraer toda la lógica que no sea mostrar la vista o componentes. Dependen de abstracciones.
- Centralizar cambios de estado: son los encargados de recibir los cambios que producen los cambios de estado. El inconveniente es que, al mostrar el estado del componente, se muestran también los widgets dependientes (todos los hijos) aunque no se utilicen.

- Mapear el formato de la vista (widget): aquí se formatean los datos que se muestran en la vista.

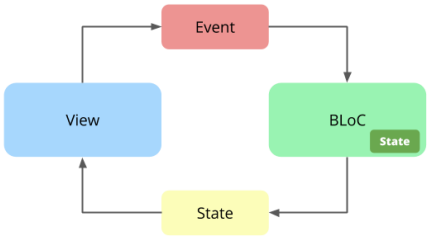

Ilustración 8. Arquitectura del patrón BLoC. Fuente: http://xurxodev.com/introduccion-al-patron-bloc/

Como se ha mencionado anteriormente, existen dos tipos de Widgets en Flutter. En ambos, es altamente recomendable utilizar este patrón. Aunque el estado sea inmutable, se favorece el rendimiento y reduce la complejidad.

Por último, hay que mencionar que existen dos tipos de implementaciones de este patrón. Por un lado, tenemos la implementación uno-uno de entrada-salida. Es decir, que cada componente de la vista esté asociado a un BLoC, reciba un evento, cambie el estado y se notifique este cambio en la vista.

Ilustración 9. Patrón BLoC, 1 entrada - 1 salida. Fuente: http://xurxodev.com/introduccion-al-patron-bloc/

Por el contrario, la implementación varios-varios, es decir, varias entradas-varias salidas, permiten asociar varios componentes al patrón sobre una misma vista, de manera que el estado de esta cambia en función del componente utilizado.

Ilustración 10. Patrón BLoC, varias entradas - varias salidas. Fuente: http://xurxodev.com/introduccion-al-patron-bloc/

Como se puede ver, se trata de un patrón muy versátil y que puede utilizarse en una gran variedad de proyectos, independientemente de su complejidad. Si bien es cierto que cuanto mayor sea esta complejidad, resultará algo más costosa su implementación. Aun así, encaja bastante bien con *Clean Architecture*, siendo esta la arquitectura recomendada y utilizada en este proyecto.

4.2.2 API Rest Servidor

Para la parte del servidor, la API se ha construido de la manera más adaptable y escalable posible. Aunque es cierto que esta implementación está pensada para usarse con una base de datos de Mongo, y por ello con las herramientas específicas para ello, si en el futuro se decidiese cambiar la base de datos, la idea es que solo haya que modificar los controladores pertinentes, o en este caso, crear nuevos controladores acordes a la base de datos que se quisiera conectar.

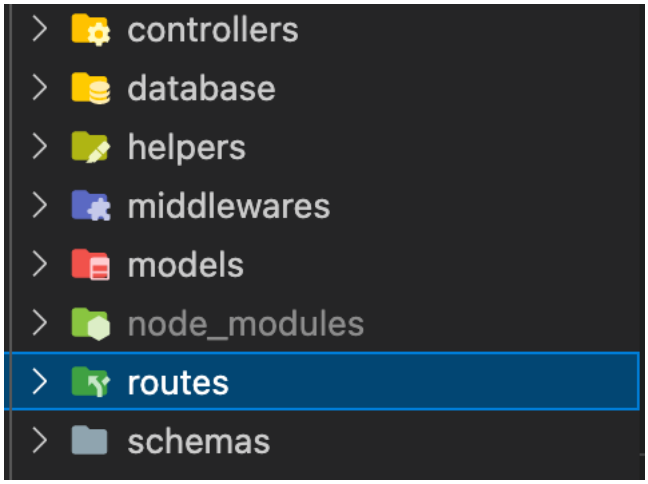

Ilustración 10. Estructura del Servidor

Como se puede apreciar en la Ilustración 11, la estructura del servidor también se divide a su vez en capas, de manera que dentro del Servidor se estaría aplicando también la arquitectura por capas descrita en la sección 4.2. Siguiendo el orden de ejecución, simulando una petición, la entrada en la aplicación sería por la carpeta *routes*, que son las rutas por las que se van distribuyendo las distintas peticiones. Una vez en la ruta correspondiente, se realizan todas las validaciones pertinentes, que se encuentran dentro de la carpeta *middlewares*. Estas validaciones variarán en función de la petición y lo que el usuario haya solicitado. En la carpeta *helpers* se encuentran algunos métodos auxilares que son comunes en varias ubicaciones del proyecto. Por último, la carpeta *controllers* cuenta con los controladores específicos de cada ruta y donde se realizan las operaciones directas hacia la base de datos, una vez que la petición ha cumplido todos los requisitos establecidos.

Hay que mencionar también, que en la carpeta *models* se encuentran los modelos de los datos que van a ser almacenados en la base de datos. En este caso, por un lado, estaría el modelo de Usuario y por otro lado los modelos de los distintos datos de actividad y ejercicio físico que se recogen desde la aplicación.

Para finalizar, en la carpeta *database* estaría recogida la configuración de la base de datos que se fuese a utilizar, en el caso de este proyecto, MongoDB.

- **Seguridad y Token de Sesión**

Uno de los aspectos fundamentales en todas las aplicaciones, tanto actuales como futuras, es la seguridad y el uso de los datos de usuarios.

Para el control de acceso y registro de usuarios, se ha decidido utilizar Firebase como herramienta encargada de estas gestiones. Por un lado, al utilizar esta herramienta se asegura no incluir información sensible (contraseña, o, mejor dicho, el hash de la contraseña) en la base de datos usada para el proyecto. Además, permite integrar de manera sencilla aplicaciones de terceros para el registro e inicio de sesión, como pueden ser proveedores como Google, Facebook, Apple, etc.

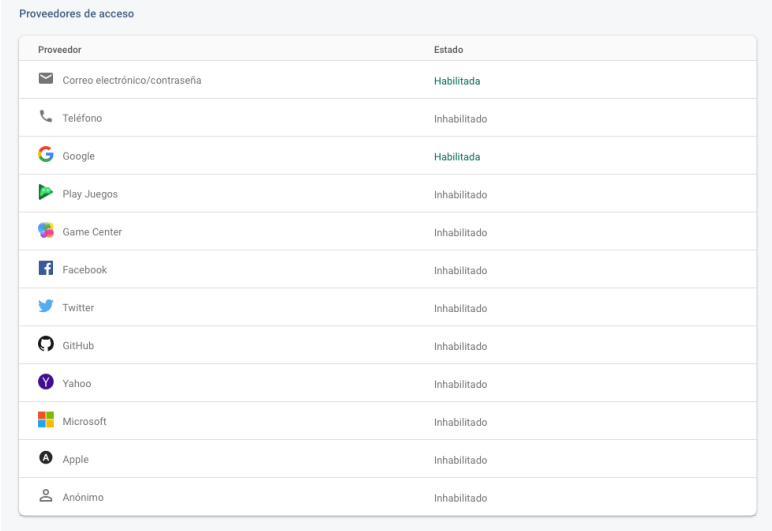

Ilustración 11. Proveedores de acceso en Firebase

Por último, permite de manera sencilla dos aspectos fundamentales. El primero, cuando un usuario se registra en la aplicación, se le envía de manera automática un email de verificación, de manera que hasta que el usuario no verifica su cuenta no podrá acceder y hacer uso de esta. Por otro lado, incluye también un sistema automático para recuperar la contraseña, por lo que, si un usuario necesitara cambiarla o recordarla porque la ha olvidado, el sistema le envía un email dónde podrá hacer este cambio.

Todo esto, de manera sencilla y sin tener que interactuar ni almacenar información sensible.

Para acabar, el otro principal motivo para su uso es que, tras iniciar sesión de manera satisfactoria, el sistema devuelve un token de sesión con una duración de una hora, que puede refrescarse. Este token es enviado en la cabecera de todas las peticiones que se realizan al servidor, y es en el servidor dónde se comprueba si este token es válido. Si bien es cierto que todo esto se podría gestionar de manera directa en el servidor, de esta manera nos aseguramos de que todas las operaciones de la API tengan la verificación de este token, incluida la creación de un usuario. Sin esta herramienta, la creación de un usuario quedaría exenta de ser verificada.

Por ello, para cada petición que se realice en la base de datos, en su mayoría peticiones *GET*, el campo del token en la cabecera y el ID de Firebase del usuario, serán los campos imprescindibles.

5

Diseño y desarrollo del sistema

En esta sección se abordarán los distintos casos de uso de la aplicación expuestos en el apartado 3.1.

Como se ha mencionado en la sección anterior, todas las operaciones hacia la base de datos tendrán que verificar que el token de sesión de Firebase es válido. Se menciona aquí de nuevo y de manera genérica para no tener que especificarlo en cada caso de uso.

5.1 Registro de un nuevo usuario

La posibilidad del registro como usuario es una de las grandes ventajas de esta aplicación. Cabe recordar que la idea es poder cambiar de dispositivo móvil y/o

pulsera/ reloj inteligente, sin perder los datos recuperados y guardados por estos dispositivos, que estarán asociados a un usuario en concreto.

A continuación, se muestra un diagrama de secuencia del proceso de registro de un nuevo usuario y la ventana de la aplicación móvil.

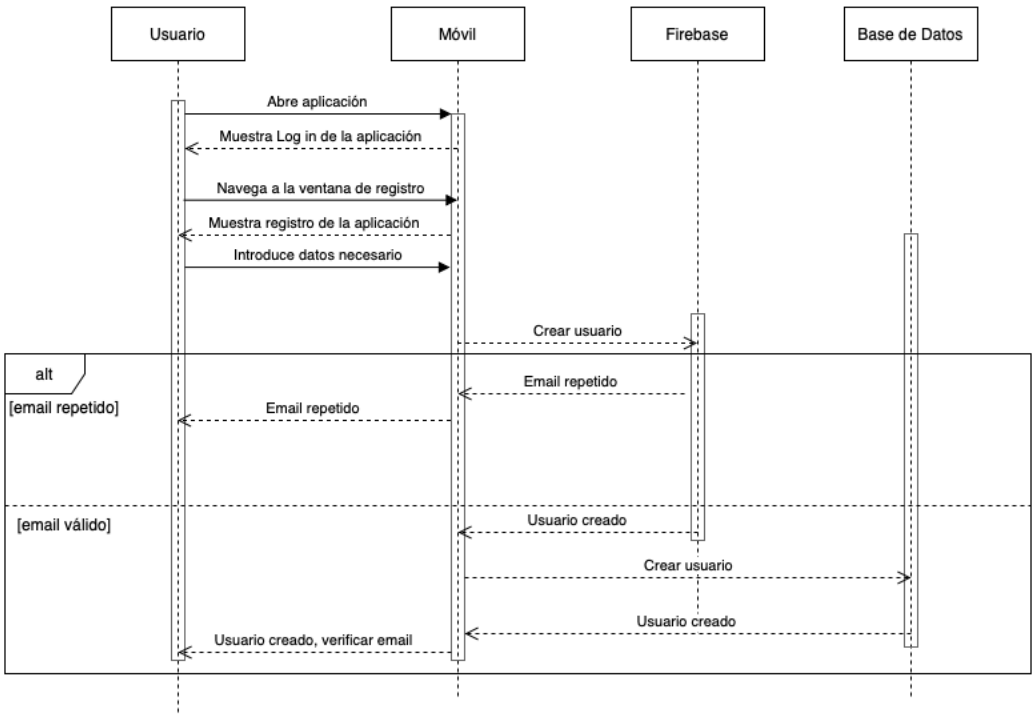

Ilustración 12. Registro de usuario

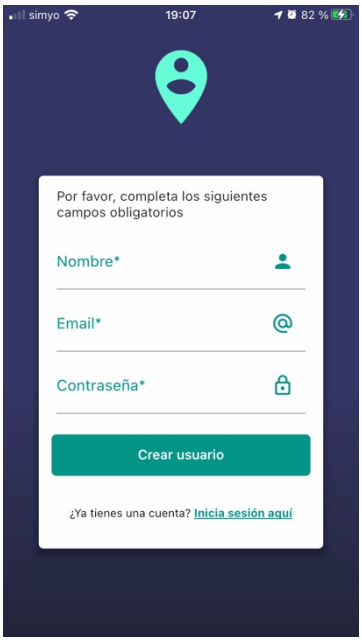

Ilustración 13. Ventana de registro de usuario de la aplicación móvil

Como se puede apreciar en la Ilustración 13, Firebase hace de intermediario para crear el usuario, siempre y cuando el email no esté repetido.

Para finalizar, se crea el usuario en la base de datos con 3 datos: nombre, email e ID generado por Firebase. Esta información podrá ser ampliada y/o modificada.

5.2 Inicio de sesión y recuperación de contraseña

Para el inicio de sesión, puramente se necesitará la interactuación con Firebase, de manera que, si las credenciales usadas para este inicio de sesión son válidas, Firebase devolverá varios datos, de los que solo harán falta el token de sesión y el ID del usuario de Firebase.

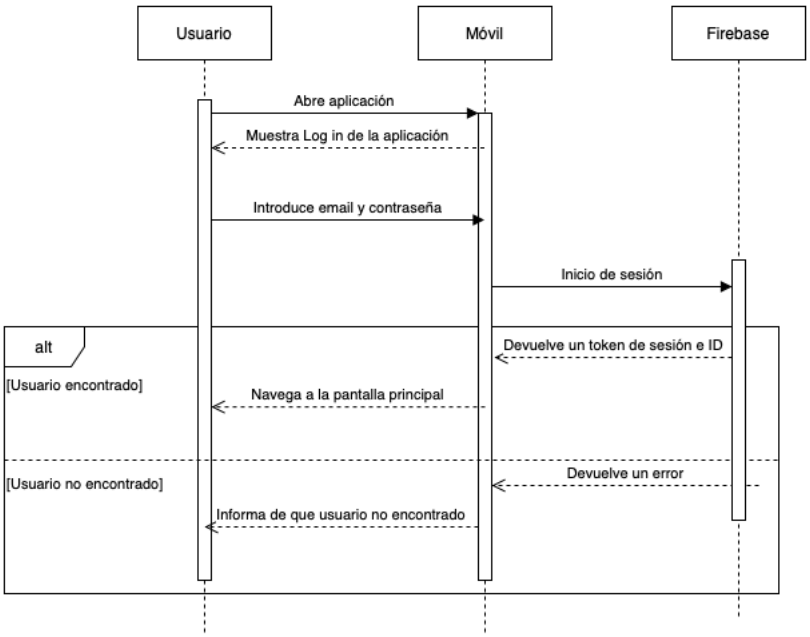

Ilustración 14. Inicio de sesión

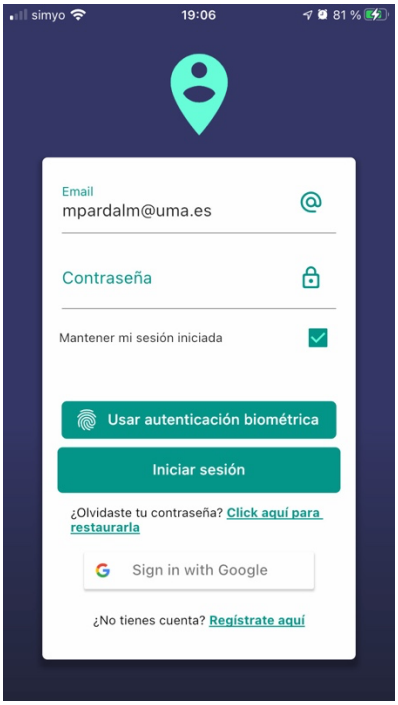

Ilustración 15. Ventana de inicio de sesión

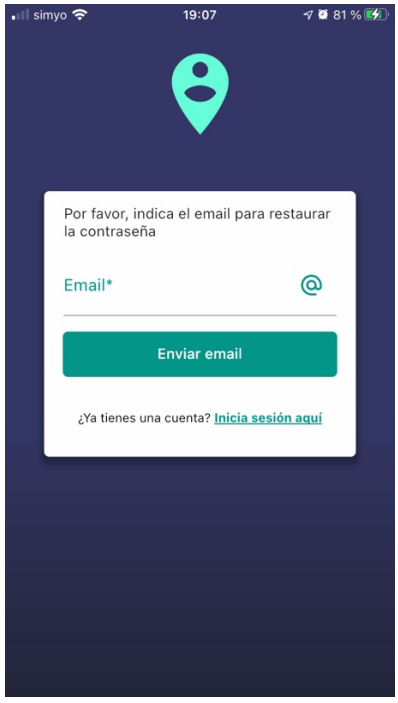

Ilustración 16. Ventana de recuperación de contraseña

En el caso de la recuperación de contraseña, si el email introducido existe en el sistema de autenticación de Firebase, se enviará un email para restaurar la contraseña.

5.3 Inicio de sesión con cuenta de Google

Como método alternativo para iniciar sesión, el usuario podrá usar una cuenta de Google para acceder a la aplicación. A partir de la autenticación de terceros, en este caso de Google, se recogerán los datos principales del usuario para almacenarlos en el sistema de persistencia y poder así vincular los datos recogidos de la pulsera/ reloj inteligente con este usuario.

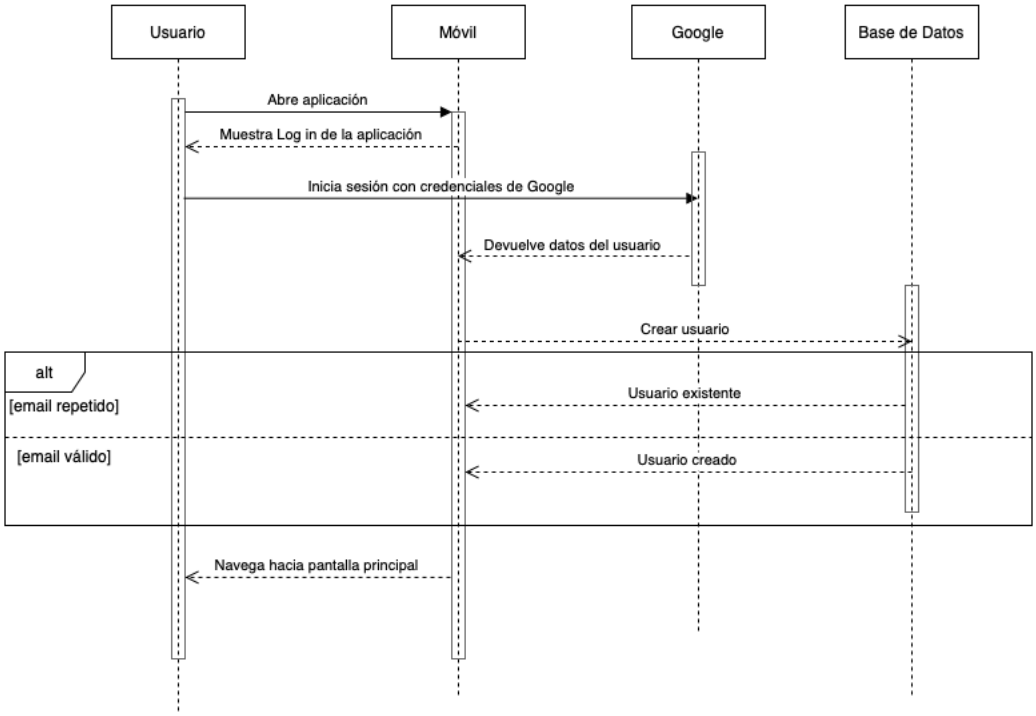

Ilustración 17. Inicio de sesión con cuenta de Google

5.4 Recogida y muestra de datos diarios principales

A partir de esta sección del proyecto, se dará por hecho que el usuario ha conseguido iniciar sesión de manera satisfactoria, por lo tanto, se omitirá en los diagramas de secuencia y su explicación.

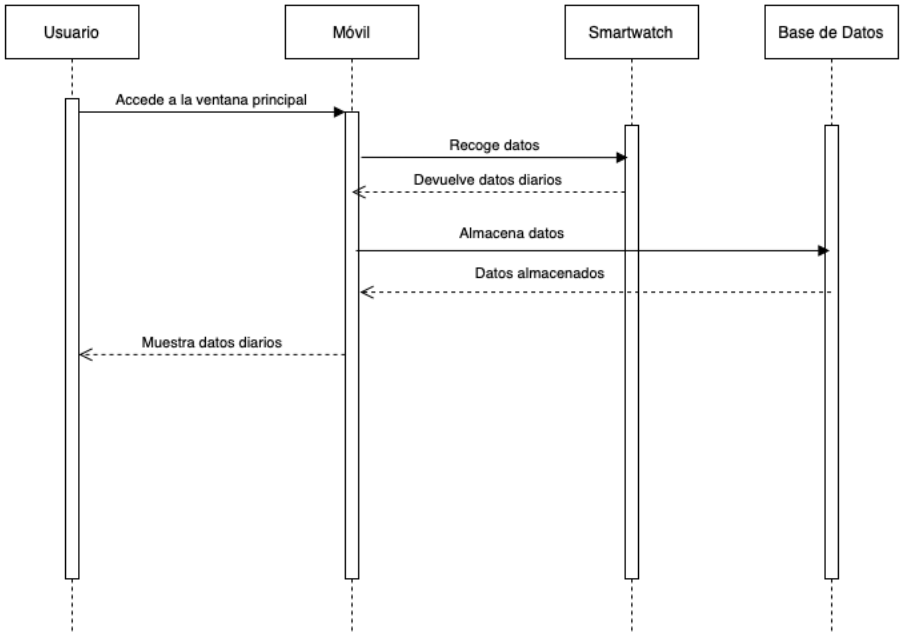

Ilustración 18. Recogida y almacenamiento de datos

En la Ilustración 18 se puede observar la secuencia para obtener y mostrar los datos diarios que son recogidos por la pulsera/ reloj inteligente. Una vez que estos datos son recogidos, se almacenan y se muestran al usuario como se pueden ver en las Ilustraciones 19 y 20.

El usuario será capaz de refrescar la información que se muestra en la pantalla deslizando esta hacia abajo. Cuando eso ocurra, el dispositivo móvil recogerá datos del reloj/ pulsera inteligente y estos se actualizarán en el sistema de persistencia.

Ilustración 20. Ventana principal 1

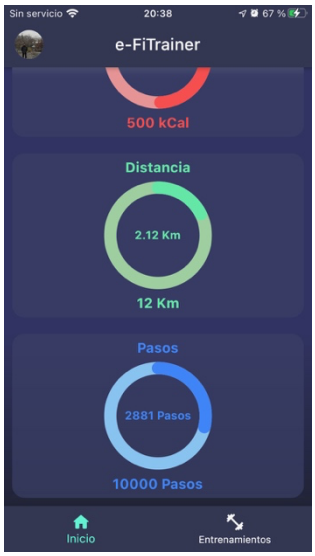

Ilustración 21. Ventana principal 2

Cada anillo representa el progreso diario de los datos recogidos. En el centro de cada uno se ve el resultado acumulado del día actual y bajo el anillo de información se encuentra el objetivo diario que se ha marcado el usuario. Este objetivo puede ser modificado por el usuario en la vista de edición de su perfil, como se verá más adelante.

5.5 Muestra de los datos filtrados por fecha mediante un gráfico

El usuario será capaz de visualizar mediante una gráfica los datos recogidos filtrándolos por una fecha determinada. Es importante destacar que se podrán ver hasta un máximo de 7 días consecutivos y no se podrán seleccionar fechas futuras a la fecha de consulta. Por ejemplo, si el usuario selecciona la fecha de inicio del 21 de mayo, se mostrarán los datos recogidos relativos desde el 21 de mayo hasta el 27 de mayo (ambos inclusive). El motivo de esto es para no sobrecargar el gráfico ni la pantalla con más datos, puesto que estos serían ilegibles.

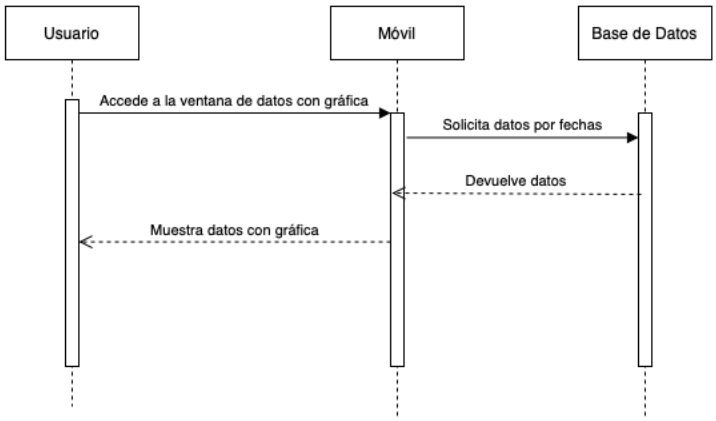

Ilustración 22. Recogida datos filtrados por fecha

Como se puede ver en las Ilustraciones 22, 23 y 24, en la parte inferior de la pantalla existen 2 recuadros. El de la izquierda muestra la media de los datos mostrados en la gráfica superior. El de la derecha muestra la media de los últimos 7 días y compara ese valor con la media de la gráfica. Si el resultado es igual o mayor, se muestra de color verde y con una flecha apuntando hacia arriba. En caso de que el resultado sea inferior, el color es rojo y la flecha apunta hacia abajo.

Ilustración 23. Detalles del movimiento

Ilustración 24. Detalles de la distancia

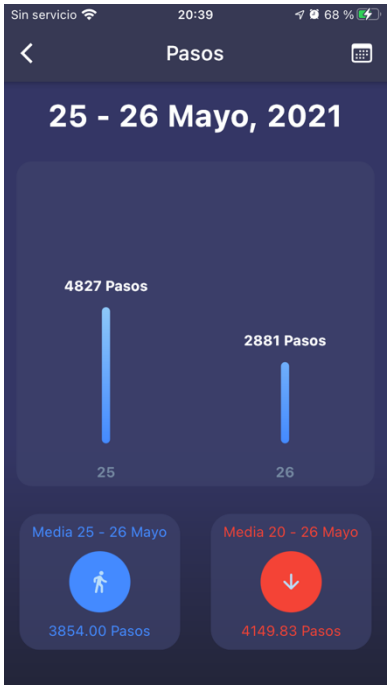

Ilustración 25. Detalles de los pasos

De manera adicional, el usuario será capaz de compartir esta pantalla con los datos que esté visualizando a través de sus redes sociales o con sus contactos.

5.6 Muestra y edición de los datos personales del usuario

El usuario podrá modificar la mayoría de los datos personales que facilita a la aplicación, a excepción del email, ya que este es un campo único en la base de datos y esencial para iniciar sesión en la aplicación.

Nombre, altura, peso, objetivo diario de kCal (movimiento), objetivo diario de distancia a recorrer y objetivo diario de pasos serán los campos que el usuario podrá modificar. Así mismo, la imagen de perfil del usuario también podrá ser actualizada.

En concreto, para actualizar la imagen de perfil se usará Firebase como sistema para almacenar esta imagen y conectarla con el usuario que ha iniciado sesión.

Respecto a los otros datos, serán almacenados en el sistema de persistencia principal de la aplicación.

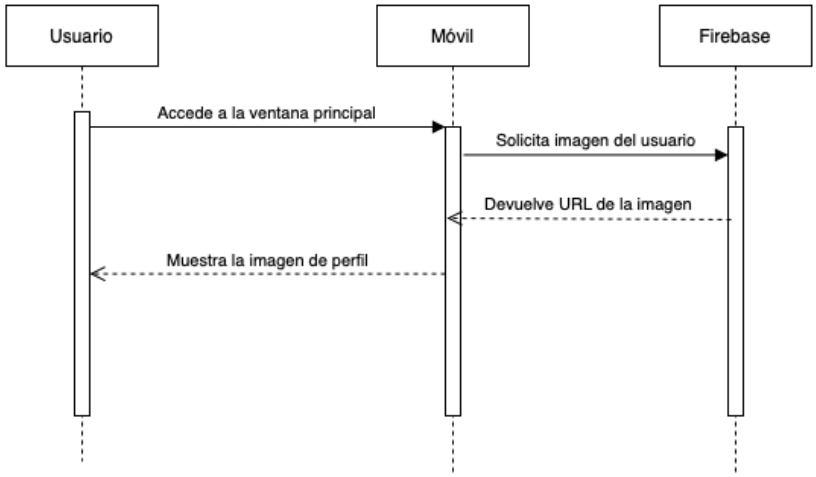

Ilustración 26. Secuencia mostrar imagen de perfil

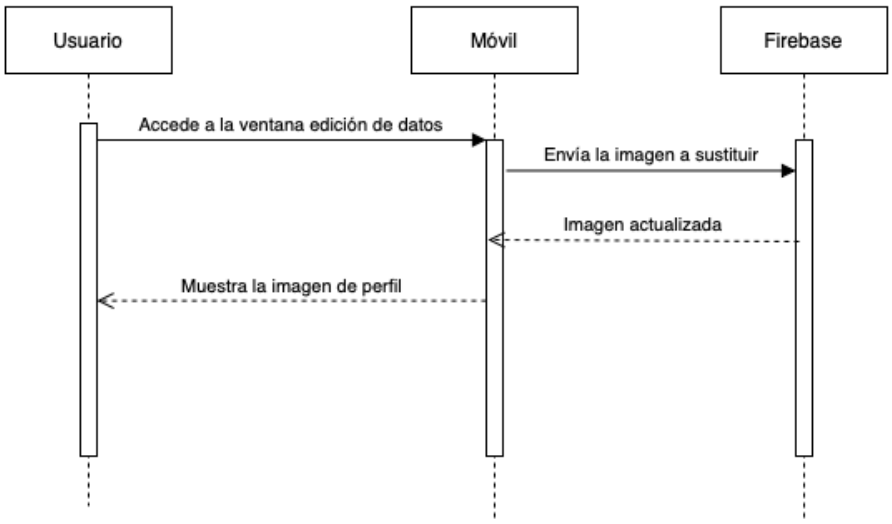

Ilustración 27. Secuencia actualización de imagen de perfil

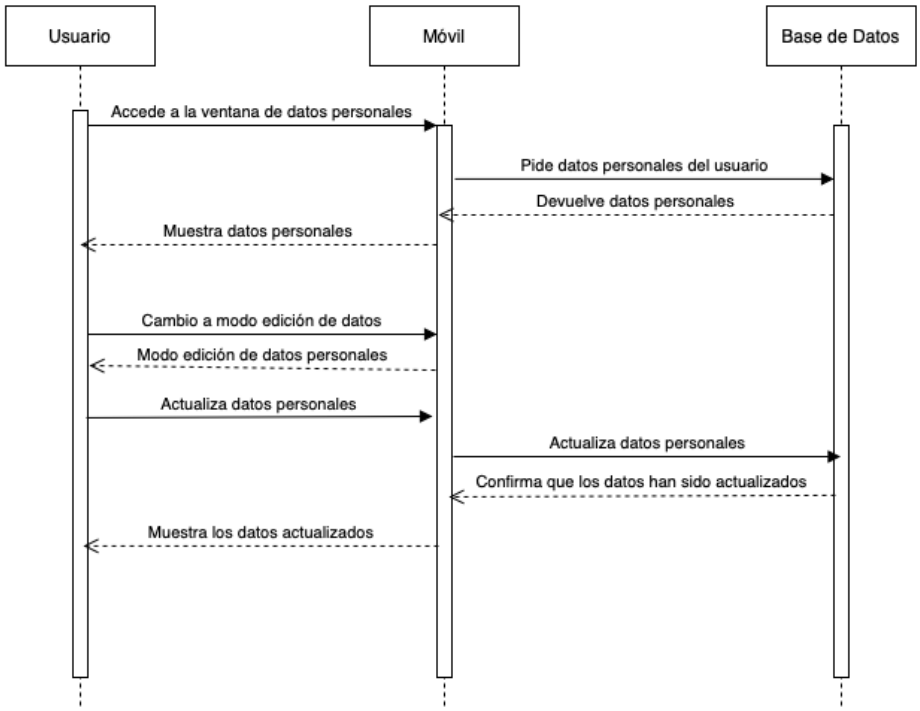

Ilustración 28. Muestra y actualización de datos personales

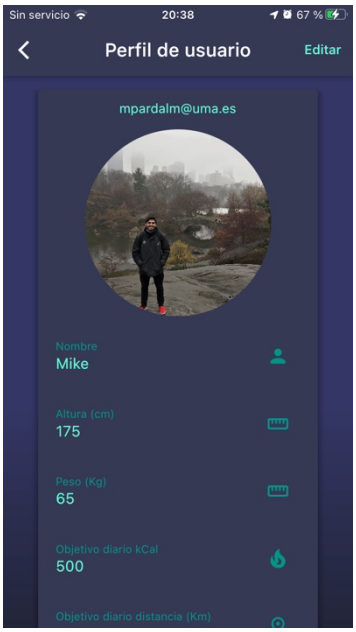

Ilustración 29. Vista de lectura de datos personales

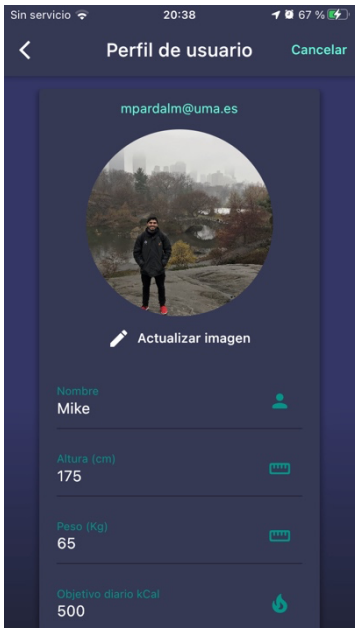

Ilustración 30. Vista modo edición de datos personales

5.7 Búsqueda y conexión de dispositivos

Desde la pantalla principal de la aplicación, se podrá en todo momento, navegar a una nueva pantalla donde aparecerán los dispositivos disponibles y aquellos a los que el usuario se podrá conectar. Una vez que el usuario se conecta a un nuevo dispositivo, la aplicación se redirigirá a la ventana principal de la misma y mostrará los datos recogidos en ese momento.

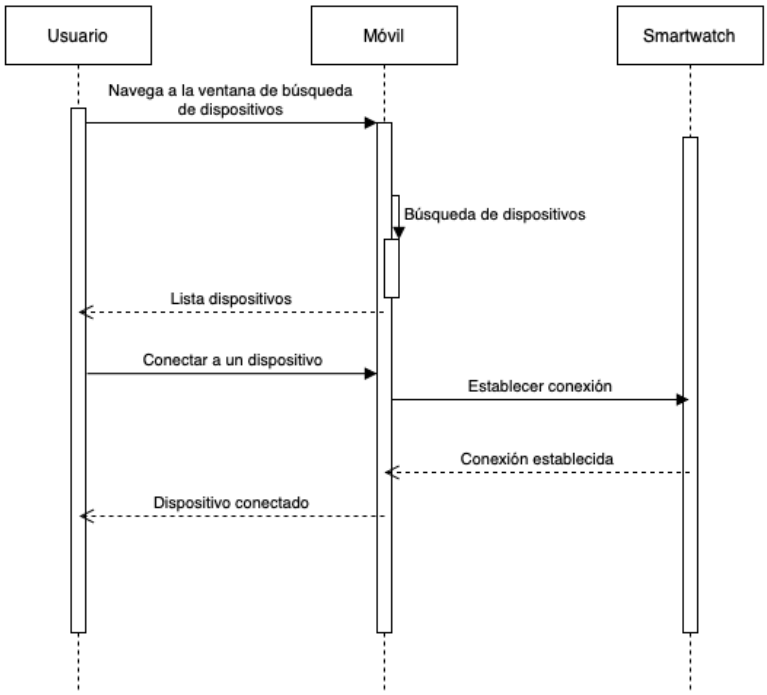

Ilustración 31. Secuencia conexión con dispositivo

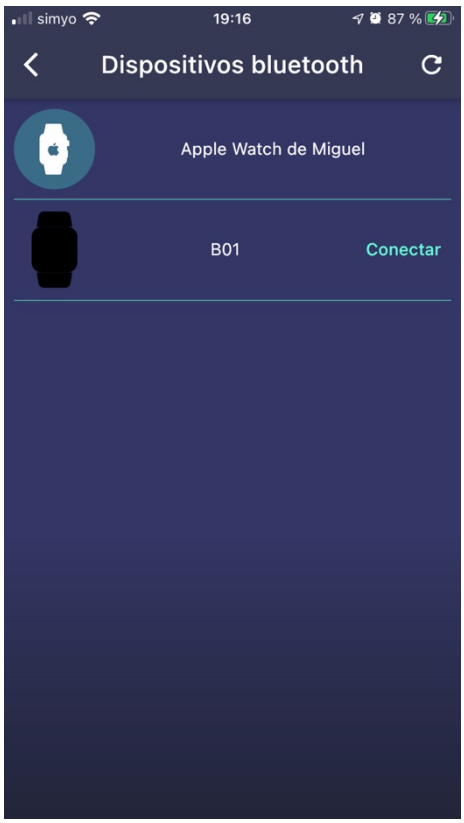

Ilustración 32. Búsqueda de dispositivos

5.8 Entrenamientos recomendados

Otra funcionalidad que se incluye en la aplicación es la de obtener entrenamientos recomendados para el usuario. En este caso, será el propio usuario el que accederá a esta sección y el sistema le mostrará una recomendación de entrenamiento. Cabe mencionar que solamente se trata de una recomendación y que toda actividad física debe estar supervisada por un especialista.

Por último, el propio usuario será el encargado de marcar ese entrenamiento recomendado como completado, indicando al sistema que le proporcione un nuevo entrenamiento.

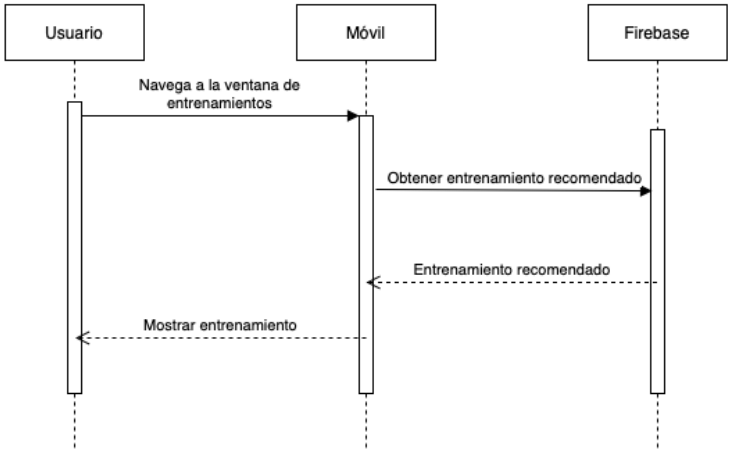

Ilustración 33. Secuencia entrenamiento recomendado

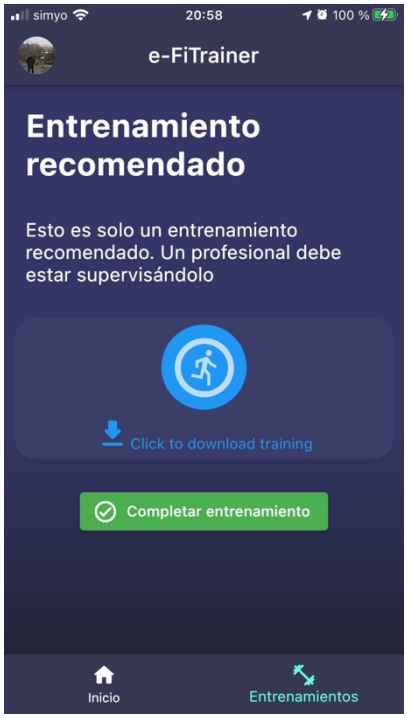

Ilustración 34. Entrenamiento recomendado

6

Pruebas y mantenimiento

Para la realización de pruebas del software desarrollado se han usado las librerías de Mocha [18] y Chai [19] en el lado del servidor, y la librería interna de Flutter para la aplicación en sí.

Empezando por las pruebas de la aplicación móvil, se realizaron test unitarios a funciones que actuaban de manera aislada. Esto es así, porque el proyecto no puede funcionar de manera independiente, es decir, la aplicación móvil necesita del servidor. Es por ello, que el grueso de los tests se encuentra en el servidor, dónde se realizarán tests *End-to-End*.

Como se puede ver en la Ilustración 31, se realizaron pruebas en funciones específicas para determinar la validez o no de los campos de contraseña y email (cumpliendo los requisitos necesarios), así como la conversión del mes en el idioma correcto.

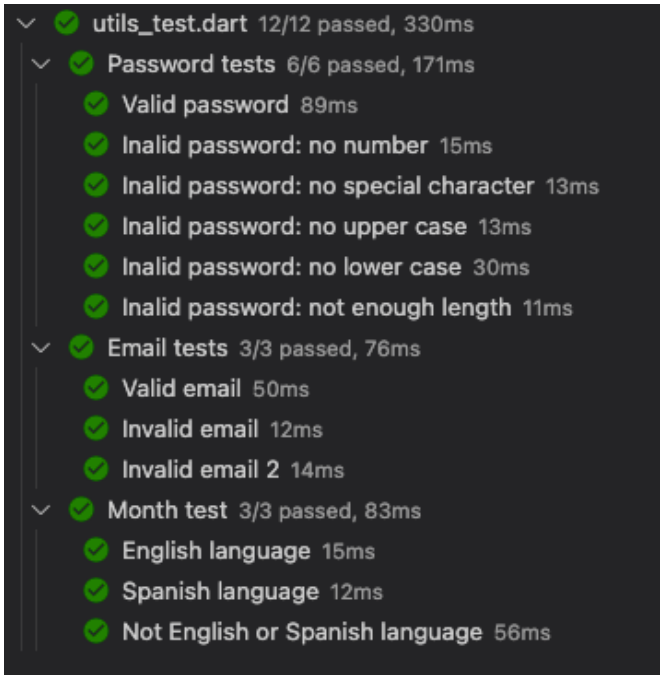

Ilustración 35. Tests unitarios en la aplicación móvil

En el caso de las pruebas en el lado del servidor, estas fueron mucho más exhaustivas porque es dónde se realiza todo el procesado de la información importante. En todas ellas se validó que el *"endpoints"* existiera y fuera correcto, que existía un token para llevar a cabo la acción, que este fuera válido y por último que existiera el usuario sobre el que se estaba realizando la acción correspondiente.

En cuanto a las acciones que se pueden realizar sobre los datos de un usuario, actualizar sus datos personales, se realizaron pruebas introduciendo datos validos y no válidos para determinar si se controlaban de manera correcta. De igual forma, en los datos recogidos por la pulsera/ reloj inteligente, se introdujeron datos no válidos para el sistema para comprobar si estos estaban controlados. En la Ilustración 32 se puede ver un resumen de estos tests.

Ilustración 19. Tests End-to-End del servidor de la aplicación

Para el mantenimiento del software, como se comentó al inicio del proyecto documento se utilizó Git como sistema controlador de las versiones del proyecto y GitHub como almacenamiento de repositorios, dado que tanto el servidor como la aplicación móvil cuentan con su propio repositorio. De esta manera, el código del proyecto está a salvo para poder ser utilizado y continuar con su desarrollo en otro equipo o con otras circunstancias.

Por último, es necesario hacer una mención a los dispositivos encargados de recopilar los datos del usuario, Apple Watch Series 3 y Xiaomi Miband 5. El dispositivo principal de recogida de datos fue el Apple Watch, dejando la pulsera Xiaomi para comprobar la integración con otras pulseras/ relojes inteligentes.

Dado el hermetismo de los productos de Apple, resultó complicado establecer la conexión de manera directa entre el reloj inteligente y la aplicación desarrollada. Por ello se optó por usar el paquete de Flutter *health* [20]. Este paquete permite acceder a los datos almacenados en el teléfono (datos relativos a actividad física y salud). De manera que, indirectamente, podemos acceder a los datos recogidos desde el Apple Watch debido a la conexión que se establece por defecto entre este reloj inteligente y un móvil iPhone.

Una vez que los datos deseados son recogidos, estos se procesan de igual forma independientemente de la pulsera/ reloj inteligente.

Respecto al dispositivo móvil utilizado para ejecutar la aplicación, se usó un iPhone 7.

7

Conclusiones y líneas futuras

En líneas generales, se ha construido una aplicación sencilla, intuitiva y fácil de usar, que cumple con sus principales propósitos: recoger información básica diaria del usuario y recomendar entrenamientos simples al usuario que use la aplicación, basándose en los datos de este.

Dada la situación mundial actual, cada vez más personas están empezando a realizar actividad física de manera independiente, ya sea en sus casas o al aire libre. La tendencia de acudir a un gimnasio para estar en forma está cambiando y es con aplicaciones como esta, como se puede facilitar recursos e información a todos los usuarios que no dispongan de ciertos recursos o simplemente no quieran acudir a un lugar cerrado a practicar ejercicio. Además, el hecho de recopilar información diaria de la actividad física realizada ayuda a que esta recomendación de entrenamientos sea lo más personalizada posible.

Teniendo en cuenta lo anterior, existen numerosas líneas futuras que pueden ser desarrolladas, aunque a continuación enumeraré solamente algunas opciones:

1) Integración con otros dispositivos, tanto relojes/ pulseras inteligentes (hay que recordar que en el presente proyecto solamente se tienen como referencia un Apple Watch Series 3 y una pulsera Xioami Mi band 5), como otros dispositivos más específicos de determinados ejercicios o actividades.
2) Integración con una inteligencia artificial, que, tras recopilar todos estos datos, los analice y vaya aprendiendo sobre la forma de actuar del usuario y las recomendaciones cada vez sean más personalizadas.
3) Recopilación de más datos. En el presente proyecto, se recopilan datos de carácter general y que sean comunes a las dos fuentes de obtener información. En función del dispositivo conectado, se podrían recopilar diferentes datos.
4) Actualización de los datos recogidos en segundo plano, de manera que, aunque la aplicación no esté abierta, cada ciertos minutos, se actualizaran los datos recopilados. Actualmente, los datos solo se actualizan cuando el usuario está interactuando con la aplicación abierta.
5) Colaboración conjunta con estudiantes de últimos años del Grado en Ciencias de la Actividad Física y del Deporte. Dado que el proyecto conecta de manera directa ambas titulaciones, hay que recordar (al igual que se ha hecho a lo largo del presente documento) que la actividad física y su recomendación debe estar supervisada por expertos en el área, luego una colaboración entre ambas titulaciones permitiría avanzar e indagar sobre el presente tema de cara al futuro. Además, con la inclusión del Grado en Ciencias de la Actividad Física y del Deporte en la Universidad de Málaga a partir del curso 2021/2022, está colaboración está más cercana que nunca.

Referencias y Bibliografía

Referencias

- [1] 2021 MongoDB, Inc, MongoDB, https://www.mongodb.com/es

- [2] 2021 MongoDB, Inc, MongoDB, https://www.mongodb.com/products/compass

- [3] OpenJS Foundation, ES6, https://nodejs.org/en/docs/es6/

- [4] OpenJS Foundation, Node.js, https://nodejs.org/es/

- [5] npm Inc, npm, https://docs.npmjs.com

- [6] Google, Flutter, https://esflutter.dev

- [7] git, https://git-scm.com

- [8] 2021 GitHub, Inc, GitHub, https://github.com

- [9] 2021 CodeFactor, CodeFactor, https://www.codefactor.io

- [10] OPenJS Foundation, eslint, https://eslint.org

- [11] Google, Firebase, https://firebase.google.com/?hl=es

- [12] 2021 Postman, Inc, Postman, https://www.postman.com

- [13] 2021 Microsoft, Visual Studio Code, https://code.visualstudio.com/docs

- [14] Flutter (Software), 2021, https://en.wikipedia.org/wiki/Flutter_(software)

- [15] Google, Dart, https://dart.dev/guides

- [16] 2021 Facebook, Inc, https://reactnative.dev/docs/getting-started

- [17] Martin, R. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship;Robert C. Martin* (Illustrated ed.). Pearson.

- [18] OpenJS Foundation, Mocha, https://mochajs.org/

- [19] Chai Assertion Library, https://www.chaijs.com/
- [20] health, https://pub.dev/packages/health

Bibliografía

Todos los enlaces han sido consultados y recuperados de páginas web en el año 2021, año de realización del proyecto:

Escacena, J. (2021, 15 marzo). *Flutter visto con gafas de programador web.* Paradigma. https://www.paradigmadigital.com/dev/flutter-visto-con-gafas-programador-web/

Fernández, J. S. (2020, 18 junio). *Introducción al patrón BLoC.* Xurxodev | Desarrollador Android y Flutter Freelance. http://xurxodev.com/introduccion-al-patron-bloc/

Iturralde, B. O. J. (s. f.). *Arquitectura en Capas.* Reactive Programming. Recuperado 17 de junio de 2021, de https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/capas

Molina, D. (2021, 8 marzo). *¿Es Flutter el framework del futuro?* BBVA Next Technologies. https://www.bbvanexttechnologies.com/es-flutter-el-framework-del-futuro/

Mroczkowska, A. (2021, 28 mayo). *Flutter vs. React Native – What to Choose in 2021?* Droids On Roids. https://www.thedroidsonroids.com/blog/flutter-vs-react-native-what-to-choose-in-2021

Vega, C. (2019, 24 marzo). *Implementa Arquitectura a tu proyecto Flutter usando el patrón BLOC.* Medium. https://medium.com/comunidad-flutter/implementa-arquitectura-a-tu-proyecto-flutter-usando-el-patr%C3%B3n-bloc-2cb031722166

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA